AsReader®

AsReader ASR-P3xU

Android SDK Reference Guide V1.0

Asterisk Inc.

Revision History

Version	Description	Date
V1.0	Initial version	2024/1/25

Contents

Introduction 5
1 Preparation for SDK Usage 6
1.1 Import SDK 6
1.2 SDK Usage 10
2 Functions 15
2.1 AsReaderP3xU 15
2.1.1 getResultCode 15
2.1.2 connectDevice 15
2.1.3 disconnectDevice 15
2.1.4 getCurrentDevice 16
2.1.5 getState 16
2.1.6 getAction 16
2.1.7 getFirmwareVersion 16
2.1.8 getHardwareVersion 17
2.1.9 getRFModuleVersion 17
2.1.10 setEventListener 17
2.1.11 removeEventListener 17
2.1.12 inventory 18
2.1.13 readMemory 18
2.1.14 writeMemory 19
2.1.15 lock 19
2.1.16 unlock 20
2.1.17 permaLock 20
2.1.18 kill 21
2.1.19 stop 21
2.1.20 defaultParameter 21
2.1.21 getBuzzer 22
2.1.22 setBuzzer 22
2.1.23 getContinuousMode 22
2.1.24 setContinuousMode 23
2.1.25 getPowerGain 23
2.1.26 setPowerGain 23
2.1.27 getPowerGainRange 24
2.1.28 getOperationTime 24
2.1.29 setOperationTime 24
2.1.30 getInventoryTime 25
2.1.31 setInventoryTime 25

2.1.32 getIdleTime 25
2.1.33 setIdleTime 26
2.1.34 setAccessPassword 26
2.1.35 getQuerySession 26
2.1.36 setQuerySession 27
2.1.37 getSessionFlag 27
2.1.38 setSessionFlag 27
2.1.39 getQValueMin 28
2.1.40 getQValueMax 28
2.1.41 getQValue 28
2.1.42 setQValue 29
2.1.43 getSerialNumber 29
2.1.44 getReportRSSI 29
2.1.45 setReportRSSI 30
2.1.46 clearEpcMask 30
2.1.47 getEpcMaskCount 30
2.1.48 addEpcMask 31
2.1.49 getEpcMask 31
2.1.50 getFrequencyAutomatic 31
2.1.51 setFrequencyAutomatic 32
2.1.52 getLbt 32
2.1.53 setLbt 32
2.1.54 getRegion 33
2.2 AsReaderP3xUEventListener 34
2.2.1 onStateChanged 34
2.2.2 onActionChanged 34
2.2.3 onReadTag 35
2.2.4 onAccessResult 35
2.2.5 onKeyEvent 36
2.3 AsReaderP3xUManager 37
2.3.1 getInstance 37
2.3.2 onDestroy 37
2.3.3 getVersion 37
2.4 AsReaderP3xUDeviceUsbCdc 37
2.4.1 AsReaderP3xUDeviceUsbCdc 37
2.5 AsReaderP3xULbtItem 38
2.5.1 getSlot 38
2.5.2 isUsed 38
2.5.3 setUsed 38

2.5.4 getFrequency 38
2.6 AsReaderP3xULockParam 39
2.6.1 AsReaderP3xULockParam 39
2.7 AsReaderP3xUPowerRange 40
2.7.1 AsReaderP3xUPowerRange 40
2.7.2 getMin 40
2.7.3 getMax 40
2.8 AsReaderP3xUSelectMaskEpcParam 41
2.8.1 AsReaderP3xUSelectMaskEpcParam 41
2.8.2 getOffset 41
2.8.3 setOffset 41
2.8.4 getLength 41
2.8.5 setLength 42
2.8.6 getMask 42
2.8.7 setMask 42
2.9 Enum 43
2.9.1 AsReaderP3xUMaskTargetType 43
2.9.2 AsReaderP3xUSessionFlag 43
2.9.3 AsReaderP3xUResultCode 43
2.9.4 AsReaderP3xUActionState 44
2.9.5 AsReaderP3xUConnectionState 44
2.9.6 AsReaderP3xUKeyType 44
2.9.7 AsReaderP3xUKeyState 44
2.9.8 AsReaderP3xUMemoryBank 44
2.9.9 AsReaderP3xUBuzzerState 45
2.9.10 AsReaderP3xUQuerySession 45

Introduction

This manual provides the following information to developers developing Android applications using the SDK.

- How the development environment is built.
- Description of various SDK library functions.

Development tools:

- Android Studio Arctic Fox | 2020.3.1
- Android SDK 24
- Android Gradle 8.1

System requirement:

- Android 10.0+

1 Preparation for SDK Usage

1.1 Import SDK

1. Click the project file "libs" in the app folder and right-click "Open in" → "Finder" (FIG. 1-1-1).

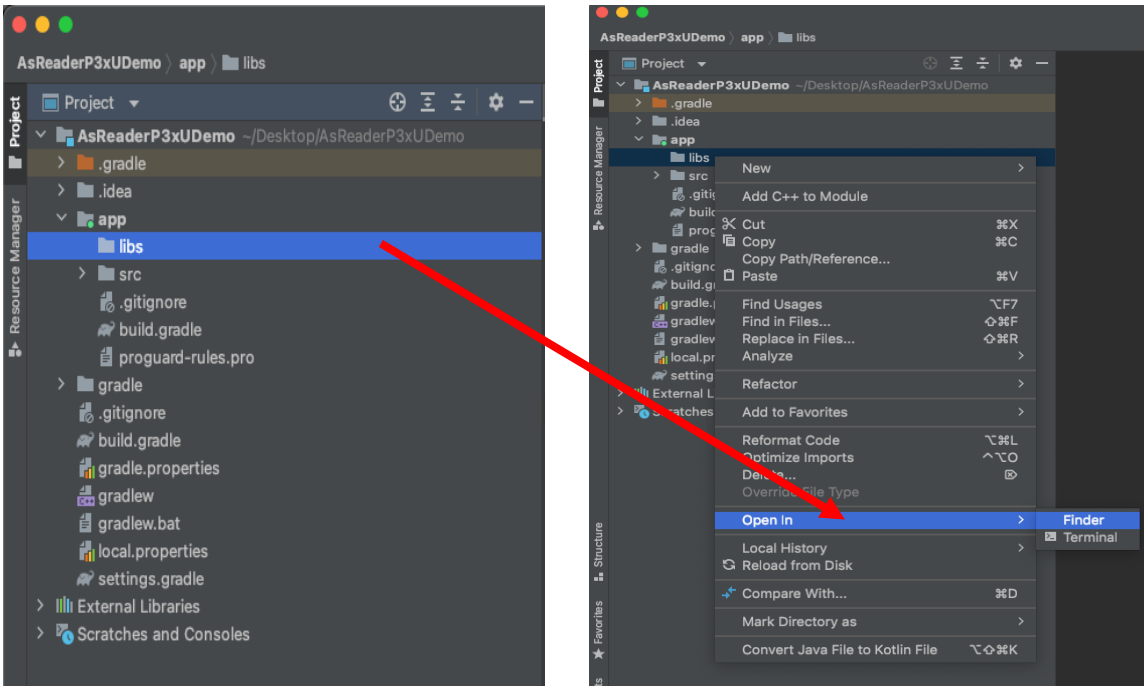

FIG. 1-1-1

2. Select the "libs" directory in the pop-up window and paste "AsReaderP3xUSDK.aar " in it (FIG. 1-1-2). After this operation is done, "AsReaderP3xUSDK.aar" will appear under "libs" of the project (FIG. 1-1-3).

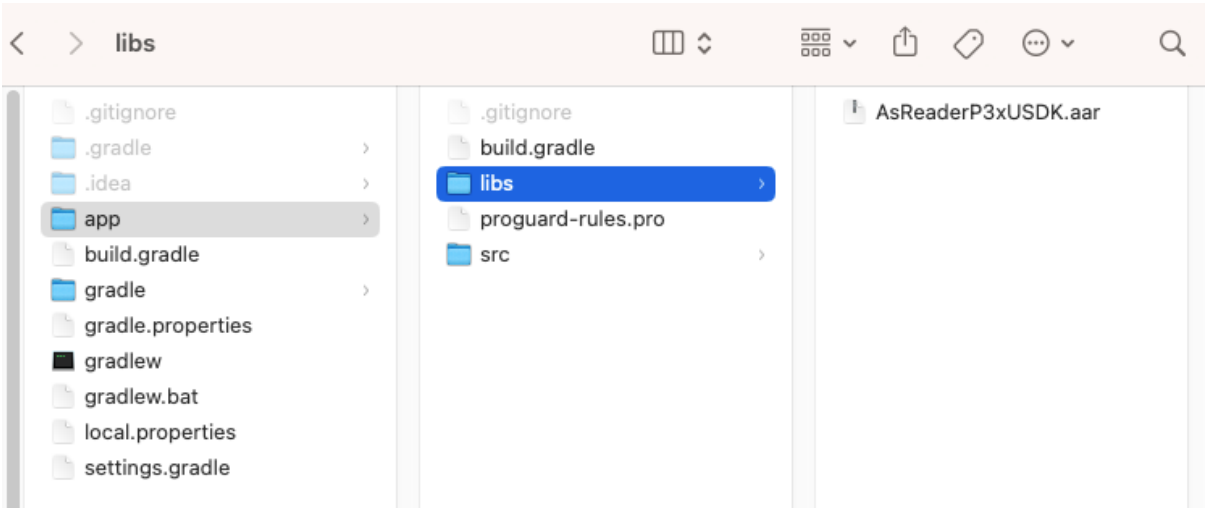

FIG. 1-1-2

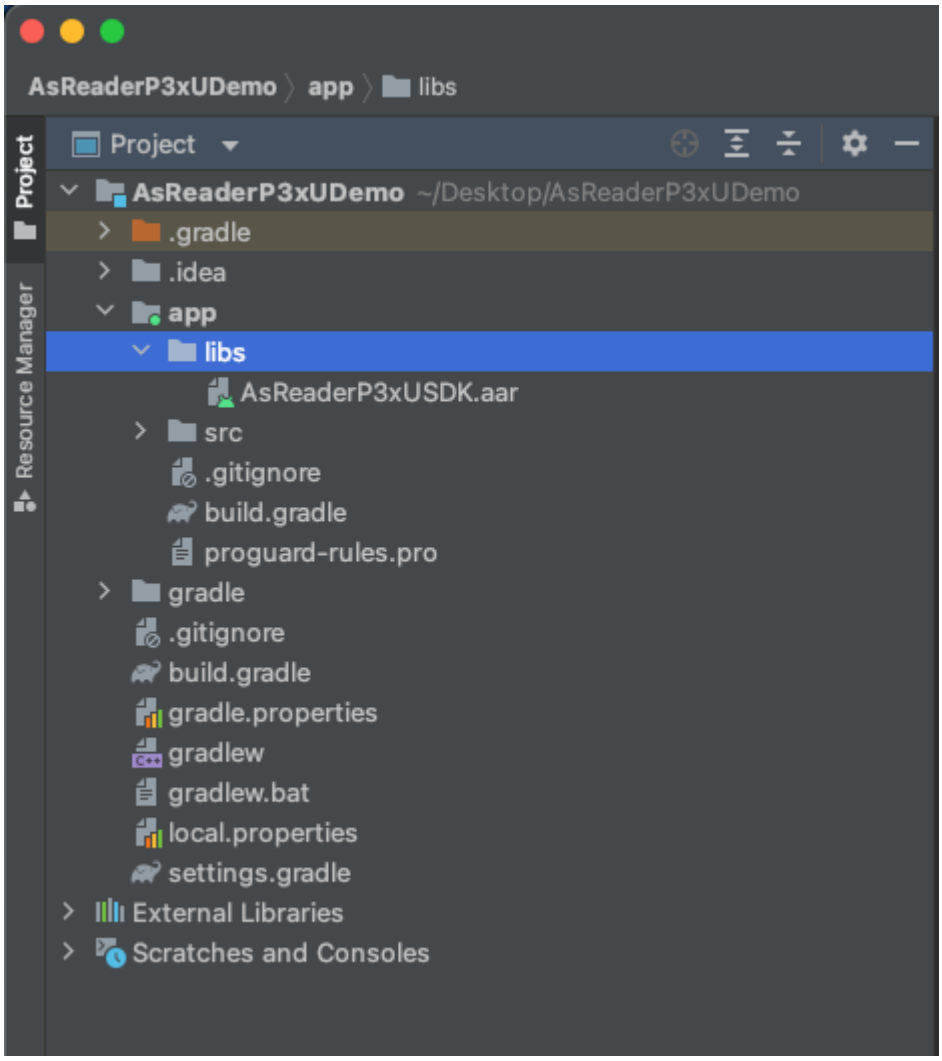
AsReaderP3xUDemo
app
libs
Project
Resource Manager
AsReaderP3xUDemo ~/Desktop/AsReaderP3xUDemo
.gradle
.idea
app
libs
AsReaderP3xUSDK.aar
src
.gitignore
build.gradle
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries
Scratches and Consoles

FIG. 1-1-3

3. Double-click to open "build.gradle" in the project (FIG. 1-1-4).

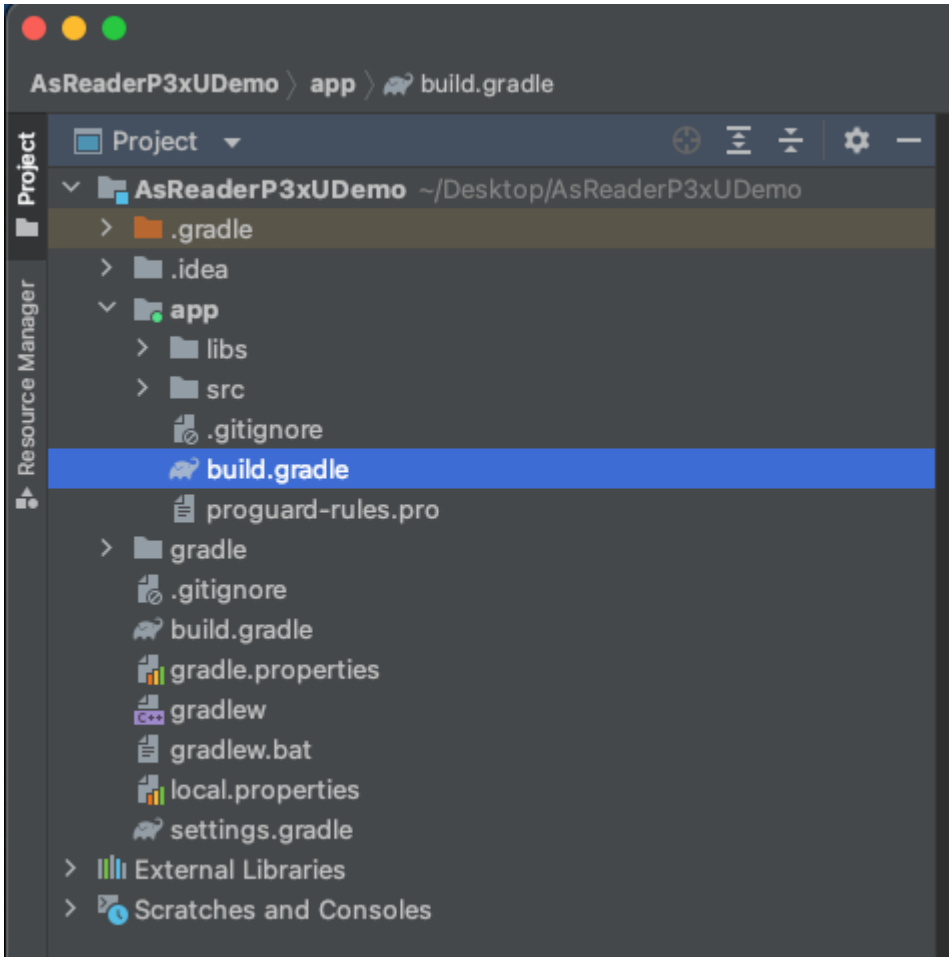

FIG. 1-1-4

4. Add the repositories and dependencies as shown in step 1 of FIG. 1-1-5 and click “Sync Now” as shown in step 2.

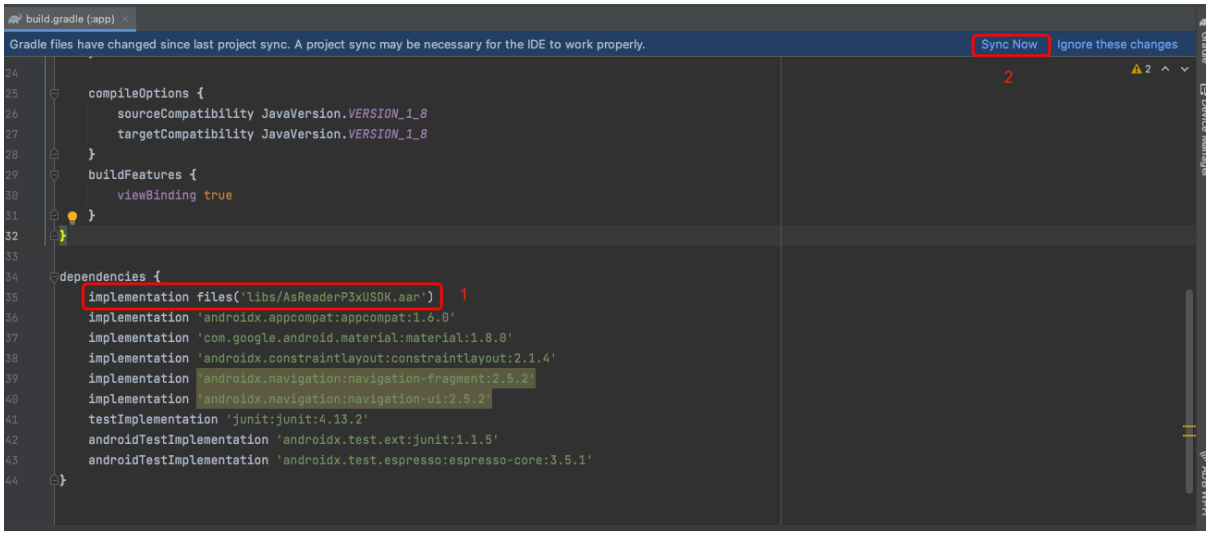

FIG. 1-1-5

5. Successful synchronization is indicated in the red box below.

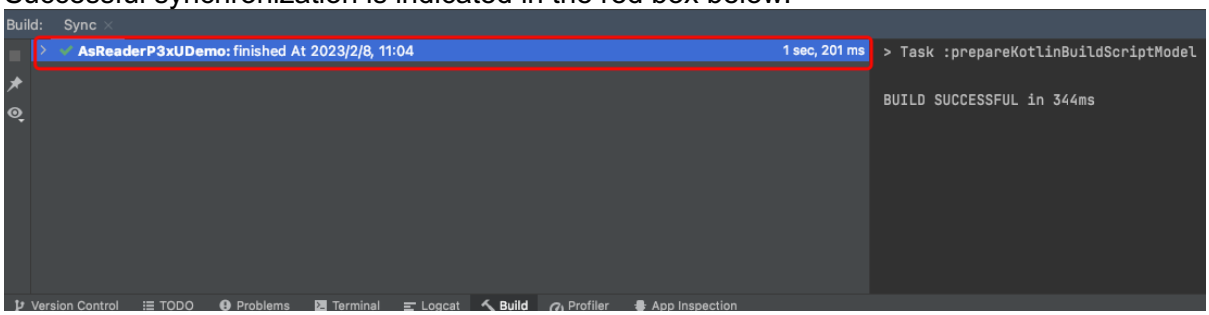

FIG. 1-1-6

1.2 SDK Usage

1. In the class to use the SDK, use the "import" statement to reference the library files (FIG. 1-2-1).

```
import com.asreader.p3xu.AsReaderP3xU;
import com.asreader.p3xu.AsReaderP3xUManager;
import com.asreader.p3xu.device.AsReaderP3xUDevice;
import com.asreader.p3xu.device.AsReaderP3xUDeviceUsbCdc;
import com.asreader.p3xu.rfid.event.AsReaderP3xUEventListener;
```

FIG. 1-2-1

2. Follow the steps below to implement the USB connection processing to the AsReader device.
   1) Implement the AsReaderP3xUEventListener
   2) Declare object
   3) Initialize the AsReaderP3xU object and set the listener
   4) USB connection

```
public class MainActivity extends AppCompatActivity implements AsReaderP3xUEventListener {

    private AppBarConfiguration appBarConfiguration;
    private ActivityMainBinding binding;

    private AsReaderP3xU mAsReaderP3xU;
    private AsReaderP3xUDevice mDevice;
    private static AsReaderP3xUDeviceUsbCdc usbCdc;
    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        setSupportActionBar(binding.toolbar);
        NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment_content_main);
        appBarConfiguration = new AppBarConfiguration.Builder(navController.getGraph()).build();
        NavigationUI.setupActionBarWithNavController( activity: this, navController, appBarConfiguration);

        usbCdc = new AsReaderP3xUDeviceUsbCdc( context: this);
        mAsReaderP3xU = AsReaderP3xUManager.getInstance();
        if (mAsReaderP3xU == null) {
            return;
        }
        mAsReaderP3xU.setEventListener(this);
        mHandler = new Handler();
        mHandler.post(mConnectDevice);
    }
    private Runnable mConnectDevice = new Runnable() {
        @Override
        public void run() {
            mDevice = mAsReaderP3xU.getCurrentDevice();
            if (mDevice == null) {
                mDevice = usbCdc;
                mAsReaderP3xU.connectDevice(mDevice);
            }
        }
    };
```

FIG. 1-2-2

3. Move the cursor to the implemented AsReaderP3xUEventListener and click "implement methods". (FIG. 1-2-3)

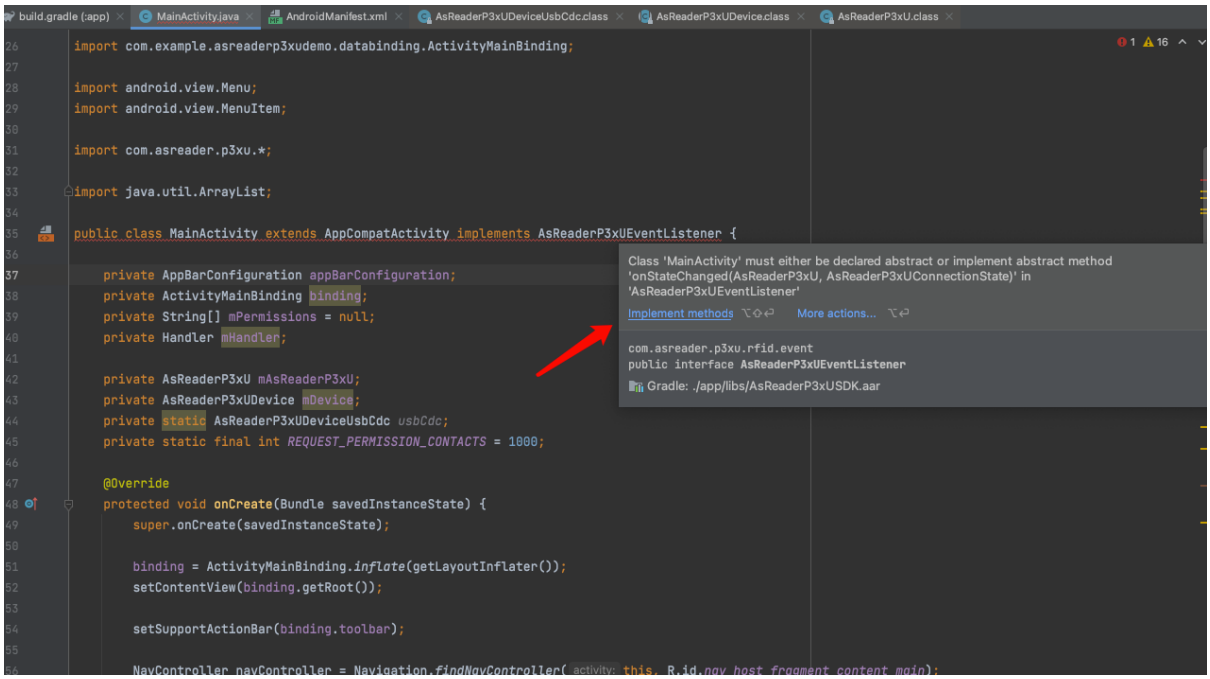

FIG. 1-2-3

4. Select the following content and click the OK button. (FIG. 1-2-4)

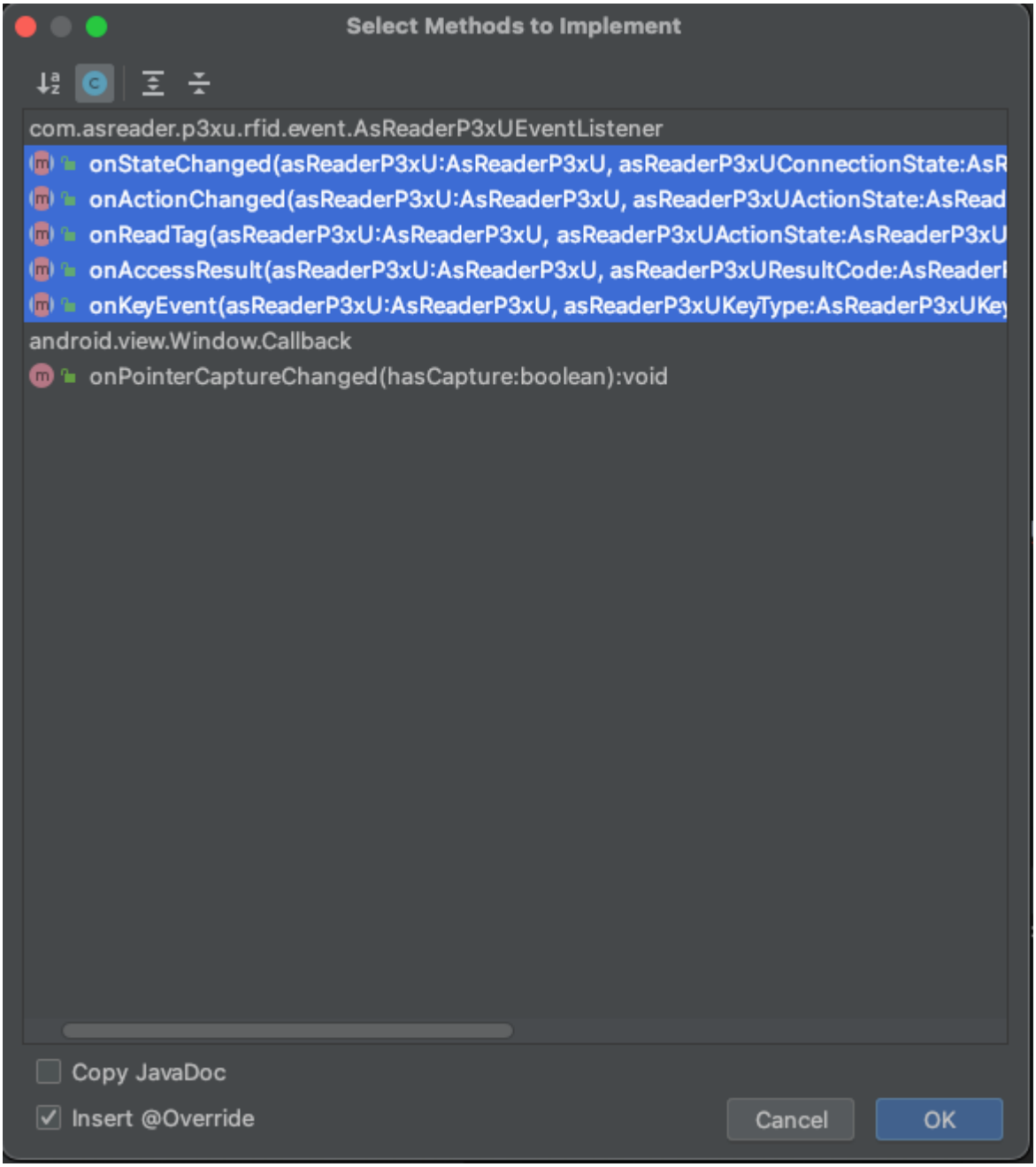

FIG. 1-2-4

5. When the OK button is clicked, the functions in the AsReaderP3xUEventListener are automatically created. (FIG. 1-2-5)

```
@Override
public void onStateChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUConnectionState asReaderP3xUConnectionState) {

}

@Override
public void onActionChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState asReaderP3xUActionState) {

}

@Override
public void onReadTag(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState asReaderP3xUActionState, String s, float v) {

}

@Override
public void onAccessResult(AsReaderP3xU asReaderP3xU, AsReaderP3xUResultCode asReaderP3xUResultCode, AsReaderP3xUActionState asReaderP3xUAction

}

@Override
public void onKeyEvent(AsReaderP3xU asReaderP3xU, AsReaderP3xUKeyType asReaderP3xUKeyType, AsReaderP3xUKeyState asReaderP3xUKeyState) {

}
```

FIG. 1-2-5

6. The connection status of the AsReader device will be returned via the function “onStateChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUConnectionState asReaderP3xUConnectionState)”. If the callback status is “Connected”, the AsReader device connection is successful.

```
@Override
public void onStateChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUConnectionState asReaderP3xUConnectionState) {
    switch (asReaderP3xUConnectionState) {
        case Disconnected:

            break;
        case Connecting:
        case Listen:

            break;
        case Connected:

            break;
        case Cancelling:
            break;
    }
}
```

FIG. 1-2-6

7. Once the connection is successful, the functions provided in the library can be called. The function “inventory ()” below is an example (FIG. 1-2-7).

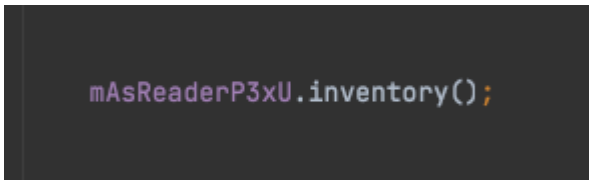

FIG. 1-2-7

8. Once function “inventory ()” is executed, the read data will be returned via the function “onReadTag”. (FIG. 1-2-8)

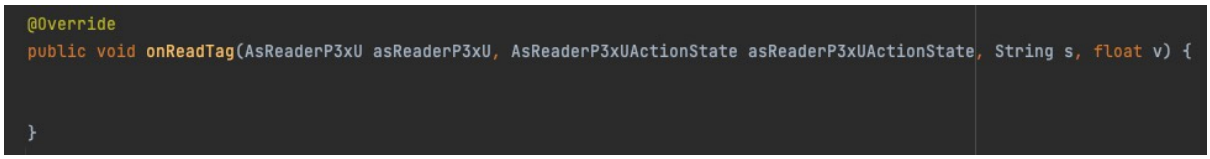

FIG. 1-2-8

2 Functions

2.1 AsReaderP3xU

2.1.1 getResultCode

Function	public AsReaderP3xUResultCode getResultCode()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUResultCode	Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Gets the result of the function execution.

■Sample code:
```
AsReaderP3xUResultCode resultcode = asReaderP3xU.getResultCode();
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.2 connectDevice

Function	public void connectDevice(AsReaderP3xUDevice device)		
Parameter	**IN/OUT**	**Type**	**Description**
device	IN	AsReaderP3xUDevice	AsReaderP3xUDevice object

■Function description:
Connects to the AsReader device.
Once this function is executed, the connection status will be returned via the function "onStateChanged" (see 2.2.1).

■Sample code:
```
asReaderP3xU.connectDevice(mDevice);
```

2.1.3 disconnectDevice

Function	public void disconnectDevice()

■Function Description:
Disconnects from the AsReader device.
Once this function is executed, the connection status will be returned via the function "onStateChanged" (see 2.2.1).

■Sample code:
```
asReaderP3xU.disconnectDevice();
```

2.1.4 getCurrentDevice

Function	public AsReaderP3xUDevice getCurrentDevice()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUDevice	AsReaderP3xUDevice object (Note: Returns null if there is no connected device.)

■Function description:
Gets the connected AsReaderP3xUDevice object.

■Sample code:
```
AsReaderP3xUDevice device = asReaderP3xU.getCurrentDevice();
```

2.1.5 getState

Function	public AsReaderP3xUConnectionState getState()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUConnectionState	Enum AsReaderP3xUConnectionState (see 2.9.5)

■Function description:
Gets the connection state of the AsReader device.

■Sample code:
```
AsReaderP3xUConnectionState connectState = asReaderP3xU.getState();
```

2.1.6 getAction

Function	public AsReaderP3xUActionState getAction()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUActionState	Enum AsReaderP3xUActionState (see 2.9.4)

■Function description:
Gets the action status of the AsReader device.

■Sample code:
```
AsReaderP3xUActionState actionState = asReaderP3xU.getAction();
```

2.1.7 getFirmwareVersion

Function	public String getFirmwareVersion()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The firmware version of the AsReader device.

■Function description:
Gets the firmware version of the AsReader device.

■Sample code:
```
String firmwareVersion = asReaderP3xU.getFirmwareVersion();
```

2.1.8 getHardwareVersion

Function	public String getHardwareVersion()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The hardware version of the AsReader device.

■Function description:
Gets the hardware version of the AsReader device.

■Sample code:
```
String hardwareVersion = asReaderP3xU.getHardwareVersion();
```

2.1.9 getRFModuleVersion

Function	public String getRFModuleVersion()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The firmware version of the RFID module.

■Function description:
Gets the firmware version of the RFID module.

■Sample code:
```
String rFModuleVersion = asReaderP3xU.getRFModuleVersion();
```

2.1.10 setEventListener

Function	public void setEventListener(AsReaderP3xUEventListener listener)		
Parameter	**IN/OUT**	**Type**	**Description**
listener	IN	AsReaderP3xUEventListener	AsReaderP3xUEventListener listener AsReaderP3xUEventListener object (see 2.2)

■Function description:
Sets AsReaderP3xUEventListener.

■Sample code:
```
asReaderP3xU.setEventListener(this);
```

2.1.11 removeEventListener

Function	public void removeEventListener (AsReaderP3xUEventListener listener)		
Parameter	**IN/OUT**	**Type**	**Description**
listener	IN	AsReaderP3xUEventListener	AsReaderP3xUEventListener listener AsReaderP3xUEventListener object (see 2.2)

■Function description:
Removes AsReaderP3xUEventListener.

■Sample code:
```
asReaderP3xU.removeEventListener(this);
```

2.1.12 inventory

Function	public AsReaderP3xUResultCode inventory()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
AsReader device starts to inventory RFID tags.
Once this function is executed, the “onActionChanged” (see 2.2.2) will be called back, returning the execution result. And the “onReadTag” (see 2.2.3) will be called back, returning the RFID tag data.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.inventory();
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.13 readMemory

Function	public AsReaderP3xUResultCode readMemory(AsReaderP3xUMemoryBank bank, int offset, int length)		
Parameter	**IN/OUT**	**Type**	**Description**
bank	IN	AsReaderP3xUMemoryBank	The memory bank of the RFID tag. Enum AsReaderP3xUMemoryBank (see 2.9.8)
offset	IN	int	The start address of the memory bank. Unit: Word
length	IN	int	The length of the memory bank to be read. Unit: Word
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
To read memory bank of the RFID tag.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.readMemory
(AsReaderP3xUMemoryBank.EPC,2,4);
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.14 writeMemory

Function	public AsReaderP3xUResultCode writeMemory(AsReaderP3xUMemoryBank bank, int offset, String data)		
Parameter	**IN/OUT**	**Type**	**Description**
bank	IN	AsReaderP3xUMemoryBank	The memory bank of the RFID tag. Enum AsReaderP3xUMemoryBank (see 2.9.8)
offset	IN	int	The start address of the memory bank Unit: Word
data	IN	String	The data to be written to the RFID tag. (Hex string)
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Writes data to the target memory bank of the RFID tag.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.
writeMemory(AsReaderP3xUMemoryBank.EPC,2, "1234");
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.15 lock

Function	public AsReaderP3xUResultCode lock(AsReaderP3xULockParam param)		
Parameter	**IN/OUT**	**Type**	**Description**
param	IN	AsReaderP3xULockParam	AsReaderP3xULockParam object (see 2.6)
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Lock a target memory bank of the RFID tag.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.lock(param);
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.16 unlock

Function	public AsReaderP3xUResultCode unlock(AsReaderP3xULockParam param)		
Parameter	**IN/OUT**	**Type**	**Description**
param	IN	AsReaderP3xULockParam	AsReaderP3xULockParam object
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Unlock the locked memory bank of the RFID tag. After unlocking, you can use the default password to overwrite the tag data.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.unlock(param);
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.17 permaLock

Function	public AsReaderP3xUResultCode permaLock(AsReaderP3xULockParam param)		
Parameter	**IN/OUT**	**Type**	**Description**
param	IN	AsReaderP3xULockParam	AsReaderP3xULockParam object (see 2.6)
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Permanently lock a target memory bank of the RFID tag.
Permanently locked tag data cannot be changed or unlocked.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.permaLock(param);
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.18 kill

Function	public AsReaderP3xUResultCode kill(String killPassword)		
Parameter	**IN/OUT**	**Type**	**Description**
killPassword	IN	String	Kill password.
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Kill a tag. The killed tag cannot be used.
Once this function is executed, the “onAccessResult” (see 2.2.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.kill("00000000");
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.19 stop

Function	public AsReaderP3xUResultCode stop()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Stop inventorying tags.
Once this function is executed, the “onActionChanged” (see 2.2.2) will be called back, returning the execution result.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.stop();
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.20 defaultParameter

Function	public AsReaderP3xUResultCode defaultParameter()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUResultCode	Function execution results. Enum AsReaderP3xUResultCode (see 2.9.3)

■Function description:
Restores default settings.

■Sample code:
```
AsReaderP3xUResultCode resultCode = asReaderP3xU.defaultParameter();
if (resultcode == AsReaderP3xUResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.21 getBuzzer

Function	public AsReaderP3xUBuzzerState getBuzzer() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUBuzzerState	Enum AsReaderP3xUBuzzerState (see 2.9.9)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the buzzer status.

■Sample code:
```
try {
      AsReaderP3xUBuzzerState mBuzzer = asReaderP3xU.getBuzzer();
} catch (AsReaderP3xUException e) {
}
```

2.1.22 setBuzzer

Function	public void setBuzzer(AsReaderP3xUBuzzerState state) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	IN	AsReaderP3xUBuzzerState	Enum AsReaderP3xUBuzzerState (see 2.9.9)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the buzzer status.

■Sample code:
```
try {
      asReaderP3xU.setBuzzer(AsReaderP3xUBuzzerState.Off);
} catch (AsReaderP3xUException e) {
}
```

2.1.23 getContinuousMode

Function	public boolean getContinuousMode() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	boolean	true: Continuous mode valid. false: Continuous mode invalid.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether the AsReader device is in inventory continuously mode.

■Sample code:
```
try {
      boolean continuousMode = asReaderP3xU.getContinuousMode();
} catch (AsReaderP3xUException e) {
}
```

2.1.24 setContinuousMode

Function	public void setContinuousMode(boolean enabled) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
enable	IN	boolean	true: Continuous mode valid. false: Continuous mode invalid.
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to inventory continuously.

■Sample code:
```
try {
      asReaderP3xU.setContinuousMode(true);
} catch (AsReaderP3xUException e) {
}
```

2.1.25 getPowerGain

Function	public int getPowerGain() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The value of power. (Unit: dBm × 10)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the power of the AsReader device.

■Sample code:
```
try {
      int powerGain = asReaderP3xU.getPowerGain();
} catch (AsReaderP3xUException e) {
}
```

2.1.26 setPowerGain

Function	public void setPowerGain(int power) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
power	IN	int	The value of power. (Unit: dBm × 10)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the power of the AsReader device.

■Sample code:
```
try {
      asReaderP3xU.setPowerGain(powerGain);
} catch (AsReaderP3xUException e) {
}
```

2.1.27 getPowerGainRange

Function	public AsReaderP3xUPowerRange getPowerGainRange() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUPowerRange	The power range can be set. AsReaderP3xUPowerRange object (see 2.7)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the power range of the AsReader device.

■Sample code:
```
try {
      AsReaderP3xUPowerRange powerGainRange = asReaderP3xU.getPowerGainRange();
      int min = powerGainRange.min; // Value of the minimum power.
      int max = powerGainRange.max; // Value of the maximum power.
} catch (AsReaderP3xUException e) {
}
```

2.1.28 getOperationTime

Function	public int getOperationTime() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	Duration of inventory. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the duration of an inventory.

■Sample code:
```
try {
    int mOperationTime = asReaderP3xU.getOperationTime();
} catch (AsReaderP3xUException e) {
}
```

2.1.29 setOperationTime

Function	public void setOperationTime(int time) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
time	IN	int	Duration of inventory. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the duration of an inventory.

■Sample code:
```
try {
    asReaderP3xU.setOperationTime(200);
} catch (AsReaderP3xUException e) {
}
```

2.1.30 getInventoryTime

Function	public int getInventoryTime() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The duration of the radio waves emitted. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the duration of the radio waves emitted when inventorying.

■Sample code:
```
try {
    int mInventoryTime = asReaderP3xU.getInventoryTime();
} catch (AsReaderP3xUException e) {
}
```

2.1.31 setInventoryTime

Function	public void setInventoryTime(int time) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
time	IN	int	The duration of the radio waves emitted. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the duration of the radio waves emitted when inventorying.

■Sample code:
```
try {
    asReaderP3xU.setInventoryTime(200);
} catch (AsReaderP3xUException e) {
}
```

2.1.32 getIdleTime

Function	public int getIdleTime() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The duration of the radio waves non-emitted. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the duration of the radio waves non-emitted when inventorying.

■Sample code:
```
try {
    int mIdleTime = asReaderP3xU.getIdleTime();
} catch (AsReaderP3xUException e) {
}
```

2.1.33 setIdleTime

Function	public void setIdleTime(int time) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
time	IN	int	The duration of the radio waves non-emitted. (Unit: ms)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the duration of the radio waves non-emitted when inventorying.

■Sample code:
```
try {
    asReaderP3xU.setIdleTime(300);
} catch (AsReaderP3xUException e) {
}
```

2.1.34 setAccessPassword

Function	public void setAccessPassword(String password) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
password	IN	String	Access password.
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the access password required for AsReader device to operate on the locked RFID tag.

■Sample code:
```
try {
    asReaderP3xU.setAccessPassword("00000000");
} catch (AsReaderP3xUException e) {
}
```

2.1.35 getQuerySession

Function	public AsReaderP3xUQuerySession getQuerySession() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUQuerySession	Enum AsReaderP3xUQuerySession (see 2.9.10)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the session value of the inventory.

■Sample code:
```
try {
    AsReaderP3xUQuerySession mSession = asReaderP3xU.getQuerySession();
} catch (AsReaderP3xUException e) {
}
```

2.1.36 setQuerySession

Function	public void setQuerySession(AsReaderP3xUQuerySession session) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
session	IN	AsReaderP3xUQuerySession	Enum AsReaderP3xUQuerySession (see 2.9.10)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the session value of the inventory.

■Sample code:
```
try {
    asReaderP3xU.setQuerySession(AsReaderP3xUQuerySession.S0);
} catch (AsReaderP3xUException e) {
}
```

2.1.37 getSessionFlag

Function	public AsReaderP3xUSessionFlag getSessionFlag() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUSessionFlag	Enum AsReaderP3xUSessionFlag (see 2.9.2)
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the session flag of the inventory.

■Sample code:
```
try {
    AsReaderP3xUSessionFlag mTarget = asReaderP3xU.getSessionFlag();
} catch (AsReaderP3xUException e) {
}
```

2.1.38 setSessionFlag

Function	public void setSessionFlag(AsReaderP3xUSessionFlag target) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
target	IN	AsReaderP3xUSessionFlag	Enum AsReaderP3xUSessionFlag (see 2.9.2)
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the session flag of the inventory.

■Sample code:
```
try {
    asReaderP3xU.setSessionFlag(AsReaderP3xUSessionFlag.AB);
} catch (AsReaderP3xUException e) {
}
```

2.1.39 getQValueMin

Function	public int getQValueMin() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The minimum Q value of the RFID module.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the minimum Q value of the RFID module.

■Sample code:
```
try {
    int mQValue = asReaderP3xU.getQValueMin();
} catch (AsReaderP3xUException e) {
}
```

2.1.40 getQValueMax

Function	public int getQValueMax() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The maximum Q value of the RFID module.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the maximum Q value of the RFID module.

■Sample code:
```
try {
    int mQValue = asReaderP3xU.getQValueMax();
} catch (AsReaderP3xUException e) {
}
```

2.1.41 getQValue

Function	public int getQValue() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The Q value of the inventory. Range: 0~15.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the Q value of the inventory.

■Sample code:
```
try {
    int mQValue = asReaderP3xU.getQValue();
} catch (AsReaderP3xUException e) {
}
```

2.1.42 setQValue

Function	public void setQValue(int value) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
value	IN	int	The Q value of the inventory. Range: 0~15.
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the Q value of the inventory.

■Sample code:
```
try {
    asReaderP3xU.setQValue(10);
} catch (AsReaderP3xUException e) {
}
```

2.1.43 getSerialNumber

Function	public String getSerialNumber() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The serial number of the AsReader device.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the serial number of the AsReader device.

■Sample code:
```
try {
    String mSerialNumber = asReaderP3xU.getSerialNumber();
} catch (AsReaderP3xUException e) {
}
```

2.1.44 getReportRSSI

Function	public boolean getReportRSSI() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	boolean	true: Gets RSSI data. false: Do not get RSSI data.
	-	AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether to get RSSI data when inventorying RFID tags.

■Sample code:
```
try {
    boolean mReportRSSI = asReaderP3xU.getReportRSSI();
} catch (AsReaderP3xUException e) {
}
```

2.1.45 setReportRSSI

Function	public void setReportRSSI(boolean enabled) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
enabled	IN	boolean	true: Gets RSSI data. false: Do not get RSSI data.
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to get RSSI data when inventorying RFID tags.

■Sample code:
```
try {
    asReaderP3xU.setReportRSSI(true);
} catch (AsReaderP3xUException e) {
}
```

2.1.46 clearEpcMask

Function	public void clearEpcMask() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	-	AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Removes EPC mask data.

■Sample code:
```
try {
    asReaderP3xU.clearEpcMask();
} catch (AsReaderP3xUException e) {
}
```

2.1.47 getEpcMaskCount

Function	public int getEpcMaskCount() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	Number of EPC masks.
		AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the number of EPC masks.

■Sample code:
```
try {
    int mCount = asReaderP3xU.getEpcMaskCount();
} catch (AsReaderP3xUException e) {
}
```

2.1.48 addEpcMask

Function	public void addEpcMask(AsReaderP3xUSelectMaskEpcParam param) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
param	IN	AsReaderP3xUSelectMaskEpcParam	AsReaderP3xUSelectMaskEpcParam object (see 2.8)
		AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Adds EPC mask.

■Sample code:
```
try {
    asReaderP3xU. addEpcMask(mask);
} catch (AsReaderP3xUException e) {
}
```

2.1.49 getEpcMask

Function	public AsReaderP3xUSelectMaskEpcParam getEpcMask(int index) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
index	IN	int	Indexs of the EPC masks.
	OUT	AsReaderP3xUSelectMaskEpcParam	Function execution results. AsReaderP3xUSelectMaskEpcParam object (see 2.8)
		AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the EPC mask for the specified index.

■Sample code:
```
try {
    AsReaderP3xUSelectMaskEpcParam mask = asReaderP3xU.getEpcMask(0);
} catch (AsReaderP3xUException e) {
}
```

2.1.50 getFrequencyAutomatic

Function	public boolean getFrequencyAutomatic() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	boolean	Whether to use frequency hopping. YES: Use. NO: Not use.
		AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether to use frequency hopping when inventorying.

■Sample code:
```
try {
    boolean mFrequencyAutomatic = asReaderP3xU.getFrequencyAutomatic();
} catch (AsReaderP3xUException e) {
}
```

2.1.51 setFrequencyAutomatic

Function	public void setFrequencyAutomatic(boolean isAutomatic) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
isAutomatic	IN	boolean	Whether to use frequency hopping. YES: Use. NO: Not use.
		AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to use frequency hopping when inventorying.

■Sample code:
```
try {
    asReaderP3xU. setFrequencyAutomatic(true);
} catch (AsReaderP3xUException e) {
}
```

2.1.52 getLbt

Function	public AsReaderP3xULbtItem[] getLbt() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xULbtItem[]	AsReaderP3xULbtItem object (see 2.5)
		AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets LBT parameters.

■Sample code:
```
try {
    AsReaderP3xULbtItem[] mItems = asReaderP3xU.getLbt();
} catch (AsReaderP3xUException e) {
}
```

2.1.53 setLbt

Function	public void setLbt(AsReaderP3xULbtItem[] table) throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
table	IN	AsReaderP3xULbtItem[]	AsReaderP3xULbtItem object (see 2.5)
		AsReaderP3xUException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets LBT parameters.

■Sample code:
```
try {
    asReaderP3xU.setLbt(table);
} catch (AsReaderP3xUException e) {
}
```

2.1.54 getRegion

Function	public String getRegion() throws AsReaderP3xUException		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The region or country the AsReader device is set to.
		AsReaderP3xUException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the region or country the AsReader device is set to.

■Sample code**:**

```
try {
      String mGlobalBand = asReaderP3xU.getRegion();
} catch (AsReaderP3xUException e) {
}
```

2.2 AsReaderP3xUEventListener

2.2.1 onStateChanged

Function	void onStateChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUConnectionState state);		
Parameter	**IN/OUT**	**Type**	**Description**
asReaderP3xU	OUT	AsReaderP3xU	AsReaderP3xU object
state	OUT	AsReaderP3xUConnectionState	Connection status. Enum AsReaderP3xUConnectionState (see 2.9.5)

■Function description:
The function will be called back when the connection status of the AsReader device changes. Once the function “connectDevice” (see 2.1.2), “disconnectDevice” (see 2.1.3) is executed, the connection status will be returned via this function.

■Sample code:
```
public void onStateChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUConnectionState state) {
   //asReaderP3xU: AsReaderP3xU object
   //state: The connection status of the AsReader device
}
```

2.2.2 onActionChanged

Function	void onActionChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState action);		
Parameter	**IN/OUT**	**Type**	**Description**
asReaderP3xU	OUT	AsReaderP3xU	AsReaderP3xU object
action	OUT	AsReaderP3xUActionState	Enum AsReaderP3xUActionState (see 2.9.4)

■Function description:
The function will be called back when the action status of the AsReader device changes. Once the function “inventory” (see 2.1.12), “stop” (see 2.1.19) is executed, the execution result will be returned via this function.

■Sample code:
```
public void onActionChanged(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState action) {
   //asReaderP3xU: AsReaderP3xU object
   //action: The action status of AsReader device
}
```

2.2.3 onReadTag

Function	void onReadTag(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState action, String tag, float rssi);		
Parameter	**IN/OUT**	**Type**	**Description**
asReaderP3xU	OUT	AsReaderP3xU	AsReaderP3xU object
action	OUT	AsReaderP3xUActionState	Enum AsReaderP3xUActionState (see 2.9.4)
tag	OUT	String	The PCEPC data of RFID tag. (Hex)
rssi	OUT	float	The RSSI data of RFID tag.

■Function description:
The function will be called back when the RFID tags are read.
Once the function "inventory" (see 2.1.12) is executed or the Trigger button is pressed, the reserved data will be returned via this function.

■Sample code:
```
public void onReadTag(AsReaderP3xU asReaderP3xU, AsReaderP3xUActionState action, String tag, float rssi) {
   //asReaderP3xU: AsReaderP3xU object
   //action: The action status of the AsReader device
   //tag: The PCEPC data of RFID tag (Hex)
   //rssi: The RSSI data of RFID tag
}
```

2.2.4 onAccessResult

Function	void onAccessResult(AsReaderP3xU asReaderP3xU, AsReaderP3xUResultCode code, AsReaderP3xUActionState action, String epc, String data);		
Parameter	**IN/OUT**	**Type**	**Description**
asReaderP3xU	OUT	AsReaderP3xU	AsReaderP3xU object
code	OUT	AsReaderP3xUResultCode	Enum AsReaderP3xUResultCode (see 2.9.3)
action	OUT	AsReaderP3xUActionState	Enum AsReaderP3xUActionState (see 2.9.4)
epc	OUT	String	The EPC data of the RFID tag. (Hex)
data	OUT	String	The read tag data. (Hex)

■Function description:
The function will be called back when function "readMemory" (see 2.1.13), "writeMemory" (see 2.1.14), "lock" (see 2.1.15), "unlock" (see 2.1.16), "permaLock" (see 2.1.17), or "kill" (see 2.1.18) is executed, returning the execution result.

■Sample code:
```
public void onAccessResult(AsReaderP3xU asReaderP3xU, AsReaderP3xUResultCode code, AsReaderP3xUActionState action, String epc, String data)  {
   // asReaderP3xU: AsReaderP3xU object
   // code: Function execution result
   // action: The action status of the AsReader device
   // epc: The EPC data of the RFID tag (Hex)
   // data: The read tag data (Hex)
}
```

2.2.5 onKeyEvent

Function	void onKeyEvent(AsReaderP3xU asReaderP3xU, AsReaderP3xUKeyType key, AsReaderP3xUKeyState state);		
Parameter	**IN/OUT**	**Type**	**Description**
asReaderP3xU	OUT	AsReaderP3xU	AsReaderP3xU object
key	OUT	AsReaderP3xUKeyType	Enum AsReaderP3xUKeyType (see 2.9.6)
state	OUT	AsReaderP3xUKeyState	Enum AsReaderP3xUKeyState (see 2.9.7)

■Function description:
Once the Trigger button of the AsReader device is pressed or released, this function will be called back.

■Sample code:

```
public void onKeyEvent(AsReaderP3xU asReaderP3xU, AsReaderP3xUKeyType key,
AsReaderP3xUKeyState state) {
   //asReaderP3xU: AsReaderP3xU object
   //key: Key type
   //state: Key state
}
```

2.3 AsReaderP3xUManager

2.3.1 getInstance

Function	public static AsReaderP3xU getInstance()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xU	AsReaderP3xU object (see 2.1)

■**Function description:**
The function used to initialize the object of AsReaderP3xU class.

■**Sample code:**
```
AsReaderP3xU asReaderP3xU = AsReaderP3xUManager.getInstance();
```

2.3.2 onDestroy

Function	public static void onDestroy()

■**Function description:**
Destroy the AsReaderP3xU object and release resources.

■**Sample code:**
```
AsReaderP3xUManager.onDestroy();
```

2.3.3 getVersion

Function	public static String getVersion()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The SDK version.

■**Function description:**
Gets the SDK version.

■**Sample code:**
```
String version = AsReaderP3xUManager.getVersion();
```

2.4 AsReaderP3xUDeviceUsbCdc

2.4.1 AsReaderP3xUDeviceUsbCdc

Function	public AsReaderP3xUDeviceUsbCdc(Context context)		
Parameter	**IN/OUT**	**Type**	**Description**
context	IN	Context	Activity that owns the DeviceUsbCdc object

■**Function description:**
The function used to initialize the object of AsReaderP3xUDeviceUsbCdc class.

■**Sample code:**
```
AsReaderP3xUDeviceUsbCdc usbCdc = new AsReaderP3xUDeviceUsbCdc(this);
```

2.5 AsReaderP3xULbtItem

2.5.1 getSlot

Function	public int getSlot()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The frequency position of the LBT frequency list.

■**Function description:**
Gets the frequency position of the LBT frequency list.

■**Sample code:**
```
int mSlot = item.getSlot();
```

2.5.2 isUsed

Function	public boolean isUsed()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	boolean	true: Valid. false: Invalid.

■**Function description:**
Gets whether the LBT is valid.

■**Sample code:**
```
boolean mIsUsed = item.isUsed();
```

2.5.3 setUsed

Function	public void setUsed(boolean used)		
Parameter	**IN/OUT**	**Type**	**Description**
used	IN	boolean	true: Valid. false: Invalid.

■**Function description:**
Sets whether the LBT is valid.

■**Sample code:**
```
item.setUsed(true);
```

2.5.4 getFrequency

Function	public String getFrequency()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The frequency when the LBT is valid.

■**Function description:**
Gets the frequency when the LBT is valid.

■**Sample code:**
```
String mFrequency= item.getFrequency();
```

2.6 AsReaderP3xULockParam

2.6.1 AsReaderP3xULockParam

Function	public AsReaderP3xULockParam(boolean killPassword, boolean accessPassword, boolean epc, boolean tid, boolean user)		
Parameter	**IN/OUT**	**Type**	**Description**
killPassword	IN	boolean	true: Lock the kill password. false: Do not lock the kill password.
accessPassword	IN	boolean	true: Lock the access password. false: Do not lock the access password.
epc	IN	boolean	true: Lock the EPC bank. false: Do not lock the EPC bank.
tid	IN	boolean	true: Lock the TID bank. false: Do not lock the TID bank.
user	IN	boolean	true: Lock the User bank. false: Do not lock the User bank.
	OUT	AsReaderP3xULockParam	AsReaderP3xULockParam object

■Function description:
Creates an AsReaderP3xULockParam object.

■Sample code:
AsReaderP3xULockParam param = new AsReaderP3xULockParam(false, false, true, false, false);

2.7 AsReaderP3xUPowerRange

2.7.1 AsReaderP3xUPowerRange

Function	public AsReaderP3xUPowerRange(int min, int max)		
Parameter	**IN/OUT**	**Type**	**Description**
min	IN	int	Value of the minimum power.
max	IN	int	Value of the maximum power.
	OUT	AsReaderP3xUPowerRange	AsReaderP3xUPowerRange object

■**Function description:**
Creates an AsReaderP3xUPowerRange object.

■**Sample code:**
AsReaderP3xUPowerRange mPowerRange = new AsReaderP3xUPowerRange(150, 260);

2.7.2 getMin

Function	public int getMin()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	Value of the minimum power.

■**Function description:**
Gets the value of the minimum power.

■**Sample code:**
int min = powerGainRange.min;

2.7.3 getMax

Function	public int getMax()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	Value of the maximum power.

■**Function description:**
Gets the value of the maximum power.

■**Sample code:**
int max = powerGainRange.max;

2.8 AsReaderP3xUSelectMaskEpcParam

2.8.1 AsReaderP3xUSelectMaskEpcParam

Function	public AsReaderP3xUSelectMaskEpcParam()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	AsReaderP3xUSelectMaskEpcParam	AsReaderP3xUSelectMaskEpcParam object

■Function description:
Creates an AsReaderP3xUSelectMaskEpcParam object.

■Sample code:
```
AsReaderP3xUSelectMaskEpcParam selectMaskEpcParam = new
AsReaderP3xUSelectMaskEpcParam();
```

2.8.2 getOffset

Function	public int getOffset()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The start address of the mask.

■Function description:
Gets the start address of the mask.

■Sample code:
```
int mOffset = selectMaskEpcParam.getOffset();
```

2.8.3 setOffset

Function	public void setOffset(int offset)		
Parameter	**IN/OUT**	**Type**	**Description**
offset	IN	int	The start address of the mask.

■Function description:
Sets the start address of the mask.

■Sample code:
```
selectMaskEpcParam.setOffset(1);
```

2.8.4 getLength

Function	public int getLength()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	int	The length of the mask.

■Function description:
Gets the length of the mask.

■Sample code:
```
int mLength = selectMaskEpcParam.getLength();
```

2.8.5 setLength

Function	public void setLength(int length)		
Parameter	**IN/OUT**	**Type**	**Description**
length	IN	int	The length of the mask.

■**Function description:**
Sets the length of the mask.

■**Sample code:**
```
selectMaskEpcParam.setLength(1);
```

2.8.6 getMask

Function	public String getMask()		
Parameter	**IN/OUT**	**Type**	**Description**
	OUT	String	The mask data. (Hex)

■**Function description:**
Gets the mask data.

■**Sample code:**
```
String mMask = selectMaskEpcParam.getMask();
```

2.8.7 setMask

Function	public void setMask(String mask)		
Parameter	**IN/OUT**	**Type**	**Description**
mask	IN	int	The mask data. (Hex)

■**Function description:**
Sets the mask data.

■**Sample code:**
```
selectMaskEpcParam.setMask(“1111”);
```

2.9 Enum

2.9.1 AsReaderP3xUMaskTargetType

Definition	Description
S0	Inventoried S0
S1	inventoried S1
S2	inventoried S2
S3	inventoried S3
SL	Selection Flags

2.9.2 AsReaderP3xUSessionFlag

Definition	Description
A	A only
B	B only
AB	A or B

2.9.3 AsReaderP3xUResultCode

Definition	Description
NoError = 0x0000	Succeed in result.
OtherError = 0x0001	An error has occurred due to unknown reason.
MemoryOverrun = 0x0003	Accessing to memory out of range.
MemoryLocked = 0x0004	Memory is locked.
NonSpecificError = 0x000F	Not a specific error.
InOperation = 0xE000	In operation.
OutOfRange = 0xE001	Out of range.
NotConnected = 0xE100	Not connected to Device.
InvalidParameter = 0xE200	Invalid parameter transmitted.
InvalidResponse = 0xE300	Returned invalid parameter.
NotSupportFirmware = 0xEE00	Unsupported firmware.
Timeout = 0xEFFF	Exceeded allowed accessing time.
OutOfRetries = 0xF009	Out of retries.
OperationFailed = 0xFFFF	Operation failed.

2.9.4 AsReaderP3xUActionState

Definition	Description
Inventory = 0x66	Inventory in progress.
ReadMemory = 0x72	Read Memory in progress.
WriteMemory = 0x77	Write Memory in progress
Kill = 0x6B	Kill Tag in progress.
Lock = 0x6C	Lock in progress.
Unlock = 0x6D	Unlock in progress.
PermaLock = 0x70	Perma Lock in progress.
Stop = 0x73	Operation Stopped.
StartDecode = 0x64	Scan Barcode in progress.
StartBuzzer = 0x75	Start Buzzer in progress.
StartVibrator = 0x76	Start Vibrator in progress.
WaitForResponse = 0xF0	Wait for response

2.9.5 AsReaderP3xUConnectionState

Definition	Description
Disconnected = 0	Disconnected
Listen = 1	Listen for connection
Connecting = 2	Connecting
Connected = 3	Connected
Cancelling = 4	Cancel the connection

2.9.6 AsReaderP3xUKeyType

Definition	Description
Trigger = 0	Trigger button

2.9.7 AsReaderP3xUKeyState

Definition	Description
KeyUp= 0	Release
KeyDown= 1	Press

2.9.8 AsReaderP3xUMemoryBank

Definition	Description
Reserved	Reserved bank
EPC	EPC bank
TID	TID bank
User	User bank

2.9.9 AsReaderP3xUBuzzerState

Definition	Description
Off	Turn off the buzzer
Low	Buzzer low
High	Buzzer high

2.9.10 AsReaderP3xUQuerySession

Definition	Description
S0	inventoried S0
S1	inventoried S1
S2	inventoried S2
S3	inventoried S3